SUPLEMENTO MENSUAL DEDICADO A LA IMAGEN SINTÉTICA

Número 23

rendermanía

Taller Virtual
Preparamos una nueva librería de casas para el mundo medieval

Informe Especial
Examinamos la última versión de Rhinoceros

El foro del lector
POV-maravillas

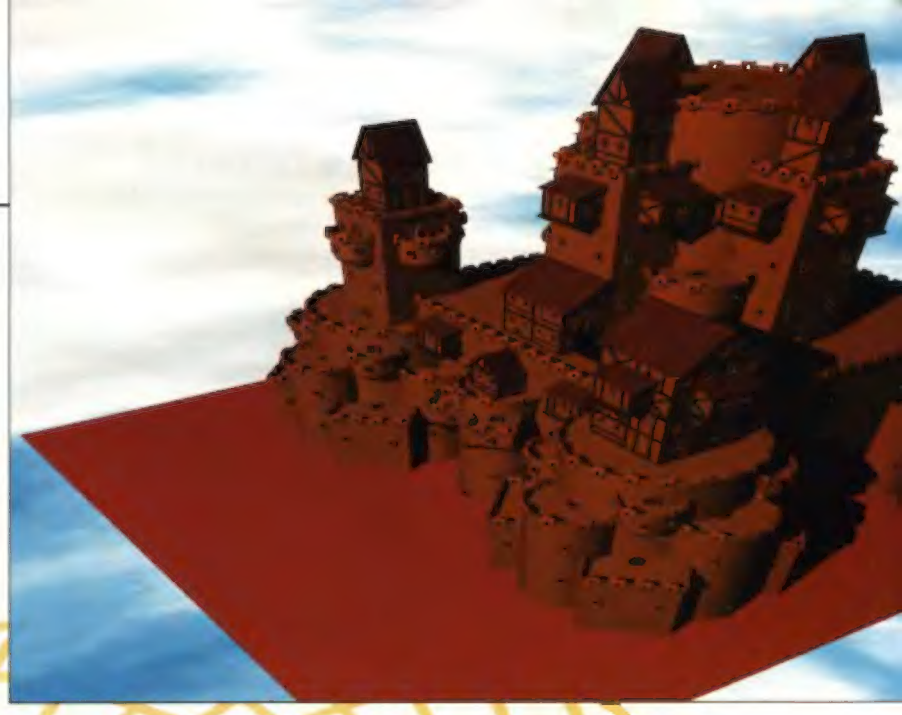
PC manía

José Manuel Muñoz

Experimentos para ciudades virtuales (parte I)

A pesar de tener como tema de sección la puesta en práctica de alguna de las ideas surgidas de las animaciones presentadas en el número anterior, las circunstancias nos obliga a preguntarnos si no sería más factible emplear la vieja librería de casas y castillos medievales publiados en los números 0 y 1. ¿Sería posible realizar alguna película con algún dragón volando sobre la ciudad o con un barco acercándose a su puerto?

Hace ya algunos números, se concibió la idea de crear la arquitectura básica de las ciudades y castillos de cada una de las razas que pueblan nuestro mundillo medieval. Por esta razón, de-



sempolvar la versión 2.0 de «Castlib» era un buen punto de partida para definir el estilo arquitectónico de las ciudades de los humanos. Como quizá recordéis, esta librería fue escrita para «POV 3.0» y estaba compuesta por objetos no demasiado complejos, cuyo propósito consistía en ayudarnos a generar castillos y ciudades medievales. Los modelos de «Castlib 2.0» son muy sencillos, pero, aunque carecen prácticamente de detalles, pueden servir para renderizar escenas bastante resultonas si se genera un número suficientemente alto de objetos utilizando bucles #while (siempre que se sitúe la cámara a una distancia adecuada).

Por otro lado, y dado que desde hace algunos números venimos dedicando artículos al tema de la animación, fue inevitable preguntarse si sería posible emplear a «Castlib» para realizar alguna película.

La lentitud de Castlib 2.0

Con esta idea en mente, se comenzó a revisar la librería para adaptarla a la versión 3.1 de «POV». Así, después de haber hecho algunos ajustes menores, se lanzaron un par de pruebas ignorando el montón de warnings con que protestó «POV» por no haber añadido los nuevos punto y coma de rigor en los sitios precisos (ver el último número de Rendermanía). El resultado fue desalentador. No por las escenas en sí, sino por la lentitud de los renders (a pesar del nuevo Pentium II a 333 MHz). ¿Por qué? Sin duda, más de un detractor de «POV» pensará que, simplemente, «POV» es lento. Pero esta respuesta debe ser descartada, ya que aquí se han generado escenas más complejas en mucho menos tiempo. Los problemas había que buscarlos en los defectos de diseño de la propia «Castlib 2.0». Estos defectos son:

- «Castlib 2.0» emplea objetos generados con el propio lenguaje escénico de «POV». Estos objetos son, en su mayoría, muy simples, pero aun así no se renderizan tan rápidamente como los objetos poligonales.
- Muchos de los objetos de la vieja librería se superponen espacialmente cuando se están construyendo casas y otros modelos. Este es el caso de las vigas de madera de los paneles básicos, por ejemplo. Tanto las casas como los castillos de «Castlib» se construyen empleando simples paneles rectangulares que incorporan ventanas, vigas y otros objetos. Al colocar estos paneles para levantar una casa, muchas de las vigas se superponen. ¿Qué que importa eso? Pues mucho en tiempo de render. En efecto, gran parte del tiempo de cálculo de cualquier trazador de rayos se emplea en calcular la intersección de los rayos lanzados con las superficies de los objetos. Pues bien, este proceso puede ralentizarse si dos o más objetos presentan superficies superpuestas.
- Los objetos que emplean CSG. Las operaciones CSG son la base de la construcción de modelos con el lenguaje escénico de «POV», por lo que no puede decirse realmente que su uso sea un error. Ocurre tan sólo que cualquier objeto poligonal se renderizará mucho más rápidamente que un objeto generado con CSG en «POV». ¿Por qué? Porque en este último caso, además de renderizar, «POV» tiene que calcular las operaciones CSG con las que se crean los objetos.

Aparte de estos problemas principales, el empleo de «Castlib 2.0» presentaba algunos inconvenientes secundarios más; crear una ciudad de 200 o 300 casas precisa bastante memoria y, si deseábamos incluir también habitantes, no había ninguna forma de evitar la superposición con otros objetos (salvo reservando algunas zonas previamente, por supuesto).

Nuevos métodos para una vieja idea

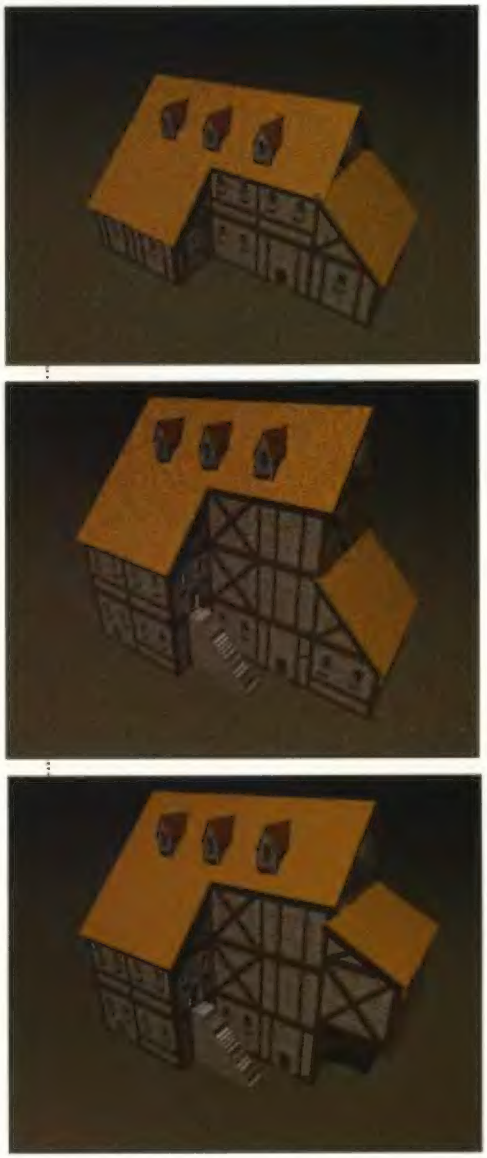
Teniendo todo esto en cuenta se optó por crear una nueva versión de «Castlib» que evitase en lo posible los defectos de la vieja librería. Después de realizar algunas pruebas y de descartar algunos diseños, se llegaron a las siguientes conclusiones:

- La idea de crear estructuras complejas como casas o torres a partir de otros objetos mucho más simples seguía siendo válida. Combinando estos

objetos básicos se pueden crear muchas estructuras diferentes, aunque inevitablemente habrá una gran similitud entre todas. Sin embargo, esto siempre será mejor que crear sólo 4 o 5 modelos complejos de casas para construir un pueblo.

Para crear las nuevas piezas básicas (los paneles, tejados, ventanas, vigas, etc.) había que utilizar un modelador poligonal. Más tarde las piezas podrían exportarse a «POV» utilizando la sentencia mesh. Esto implicaba, al menos, dos ventajas. En primer lugar, los objetos poligonales que se almacenan con mesh ocupan muy poco espacio ya que, a partir de la segunda copia, «POV» emplea matrices para referenciar al objeto original en lugar de duplicar la memoria. Y en segundo lugar, las mallas de polígonos (almacenadas con las sentencias triangle y smooth_triangle de «POV») ahorran mucho tiempo de cálculo.

¿Por qué no aprovechar las nuevas macros de «POV» y los arrays? Empleando macros es factible realizar rutinas que generen casas e incluso ciudades enteras por nosotros. En «Castlib 2.0» se "declaraban" 40 o 50 casas construyéndolas a partir de las piezas básicas. Para «Castlib 2.5», sin embargo, podíamos escribir macros que generasen las casas aleatoriamente y que colocasen escaleras o puentes siguiendo criterios más dependientes del terreno. En cuanto a los arrays, podíamos emplearlos para marcar las zonas libres y poder situar así a los personajes.



Casas definidas de la antigua librería.

Utilizar «Rhinoceros»

Aceptando estas premisas, el siguiente paso fue elegir el modelador adecuado para crear las piezas básicas. Aunque esto podía haberse hecho utilizando cualquier buen modelador poligonal, la elección finalmente recayó en Rhinoceros, por varias razones. La primera porque las piezas básicas que debían crearse eran de muy fácil construcción utilizando las fiables operaciones CSG de «Rhino», y la segunda porque «Rhinoceros» tiene exportación directa a «POV» y genera archivos donde las mallas se almacenan con las sentencias de este raytracer. Además, la opción de «Rhinoceros» de conversión de objetos nurb a mallas poligonales, que antes no estaba demasiado bien terminada, funciona ahora perfectamente.

El primer paso fue definir la escala de trabajo. Los lectores que hayan echado un vistazo a los personajes de nuestro mundo medieval sabrán que estamos utilizando una escala en la que 100 unidades equivale a un metro. Por tanto, lo primero fue adaptar el tamaño del grid y del snap a esa escala asignando un tamaño de un decímetro a cada intersección de la retícula. Así, un panel de 3 metros de alto por uno de largo ocupa 30*10 cuadrículas que equivalen a 300*100 unidades.

El siguiente paso fue crear las piezas básicas de la librería. Estas piezas son paneles rectangulares y triangulares que representan a trozos de las paredes de las casas. Casi todas estas piezas se "elevaron" a partir de curvas cerradas utilizando la opción "Extrude Planar Curve" ya explicada en el artículo anterior y con la que se crean formas sólidas. Posteriormente, algunos de estos paneles fueron recortados empleando operaciones CSG para crear los huecos pa-

ra las puertas y ventanas. Posteriormente, las piezas terminadas se almacenaron como ficheros de «Rhino» y sólo cuando se tuvo un número lo suficientemente elevado de ellas se pensó en exportarlas a «POV».

La generación de mallas

Las principales opciones de trabajo de «Rhino» están pensadas para la creación de superficies y sólidos nurb. Antes de exportar nuestro trabajo al formato de «POV» o al de cualquier otro programa, hemos de convertir en mallas los objetos nurbs con los que hemos estado trabajando. Si no lo hacemos así, el mismo «Rhino» nos presentará la ventana de conversión antes de efectuar la grabación. Emplear adecuadamente las opciones de esta ventana tiene una importancia capital ya que si nos descuidamos podemos acabar con una malla de cientos o miles de polígonos para representar algo tan sencillo como un cubo. En versiones anteriores de «Rhinoceros», esta conversión era bastante problemática y únicamente justificable en el caso de modelos de cierta complejidad como cabezas o armaduras. El problema era, en resumen, que los algoritmos de «Rhino» tendían a producir muchos más polígonos de los necesarios. Afortunadamente, este problema está arreglado en la última versión del programa.

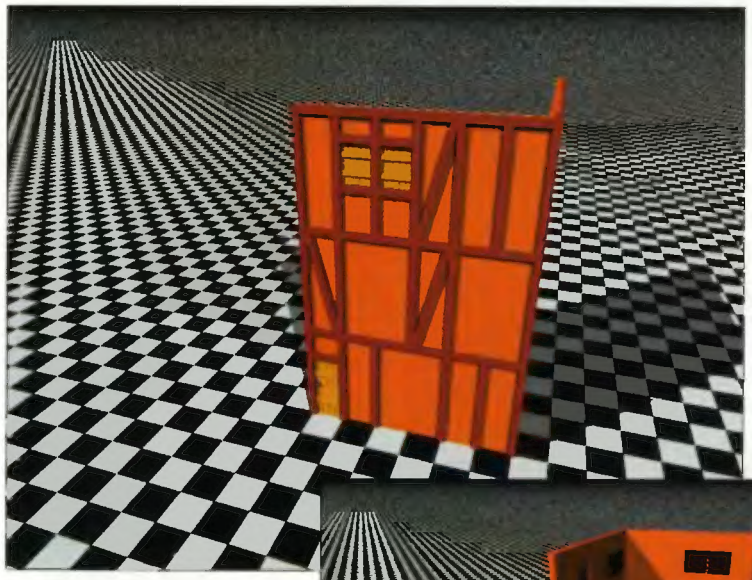
Para convertir a poligonal un objeto nurb, habrá que seleccionarlo y pinchar en el icono "mesh from Nurbs Object" situado en la barra vertical de iconos, bajo el icono de las operaciones booleanas. (Con este icono también se puede acceder a un menú emergente de iconos que permite crear objetos poligonales sencillos y realizar diversas operaciones con ellos). A continuación, se desplegará la ventana de conversión que puede aparecer en dos formatos distintos: "Simple Controls" y "Detailed Controls". En el primer caso, «Rhino» presentará una ventana muy simple donde lo único que hay que hacer es mover un indicador para indicar al programa el grado de detalle de la malla a generar. Si se desplaza el indicador hacia "Fewer polygons", «Rhino» intentará crear una malla con el menor número posible de caras. Hacer lo contrario llevando la barra hacia "More Polygons" tendrá el efecto opuesto.

Después de mover este indicador generaremos el objeto pulsando sobre OK. Aparecerán mensajes indicando los polígonos que se han generado y el objeto malla se superpondrá sobre el antiguo objeto-nurb, el cual seguirá seleccionado. Si no nos convence el resultado podemos retroceder con undo (Ctrl-Z) y volver a invocar la ventana de conversión pulsando el botón derecho del ratón sobre la ventana de trabajo (repetición de acción anterior).

Pero para tener un grado de control superior sobre la conversión será mejor, sin duda, utilizar la ventana alternativa a la que accederemos pulsando sobre "Detailed controls". En dicha ventana la conversión puede controlarse de varias formas pero lo más usual es emplear el recuadro "max angle" en el que podemos especificar el ángulo máximo que permitirá «Rhino» entre las normales de la malla a crear. (Nota: la normal de un polígono se representa con una flecha que abandona a éste de manera perpendicular a su superficie). Al generar la malla, «Rhino» irá generando triángulos subdividiendo las zonas nurb entre las que haya ángulos. Así, si entre las normales de dos triángulos adyacentes hay un ángulo superior al especificado, el algoritmo creará un triángulo intermedio. Naturalmente, cuanto menor sea el valor que empleemos más densa será la malla resultante y viceversa. También suele ser muy conveniente marcar los recuadros "Refine" y Weld".

Ahora bien, la gran mayoría de las piezas a convertir en el presente caso eran formas planas y rectangulares (frecuentemente simples cubos estira-

dos en algún eje) y por ello las opciones mencionadas resultaban imprecisas y, a veces, inútiles. En estos casos lo que se hizo fue marcar la opción "Simple Planes". Esta opción genera mallas muy optimizadas para formas simples, llegando en ocasiones a crear objetos de tan sólo 12 polígonos en el caso de algunos paneles o vigas. Recordad, no obstante, que esta opción es bastante inútil en los casos de objetos curvados (como las vigas de soporte laterales de las casas o las piezas de recorte para las ventanas de piedra, por ejemplo).



La variable lados determina los lados visibles.

Filosofía en la construcción de la librería

El criterio principal al construir las piezas de «Castlib 2.5» ha sido el de conseguir la máxima velocidad posible durante el render. Por ello, en ocasiones se han almacenado piezas que están realmente duplicadas varias veces dentro de un mismo objeto. Este es el caso de los maderos de la puerta o de los barrotes del balcón. Hubiera sido muy sencillo almacenar un solo madero o un solo barrote y generar desde «POV» el resto de las piezas necesarias para montar el objeto. Sin embargo, estas piezas tenían tan pocos polígonos que se prefirió no obrar de esta manera y ganar así algo de velocidad, por poca que fuese. Hay que decir, no obstante, que no todas las piezas están almacenadas de una manera óptima o siguiendo siempre el mismo criterio. Así, por ejemplo, aunque la gran mayoría de las piezas se han almacenado con el borde pegado al eje Y (verde) y descansando sobre el eje X (rojo), éste no ha sido siempre el caso, algunos paneles frontales se han almacenado desplazados porque en muchos casos se esperaba que quedarían en las posiciones originales de desplazamiento al ser renderizados.

En cuanto a los propios ficheros que forman la librería, hay que decir que en el momento de escribir estas líneas su organización es un poco caótica. El motivo es que «Castlib 2.5» es, todavía, una versión beta. Aunque prácticamente todas las piezas básicas están ya almacenadas en los ficheros .inc, aún faltan varias de las macros principales que nos permitirán generar las escenas. Y es que «Castlib 2.5» es una librería en parte de objetos y en parte de macros (ver el número anterior). Los objetos son las piezas básicas de construcción, pero éstas no se montarán "a mano" sino dando parámetros a las macros suministradas.

Pero por ahora centrémonos en los objetos. Éstos están almacenados en una serie de archivos de los cuales el principal es Castlib1.inc. La estructura de este fichero es muy simple, estando todas las piezas agrupadas por tamaños. Primero tenemos los trozos de pared de 3x1 metros, lue-

Comprobación de los tamaños de varios objetos de «Castlib 2.5».

go los de 3x2, etc. Generalmente, dentro de estos apartados se definen primero las vigas, luego los trozos de pared desnudos y después los trozos de paredes montadas, es decir, con las puertas, ventanas y vigas definidas previamente. Se ha procurado que los nombres de estas piezas sean tan explícitos como fuese posible y por ello tenemos identificadores como "panelR3x1_puerta10", que significa que la pieza es un panel de pared rectangular (R) que incluye la puerta de tipo 10, o como "barandilla_escalera_3x3", que se refiere a la barandilla de una escalera de 3 metros de largo (en Z) y 3 metros de subida (en Y). Por ahora, la lectura de Castlib1 es bastante fastidiosa ya que los objetos importados están intercalados con las definiciones de las piezas "completas". Además, muchos objetos-mesh demasiado grandes se han trasladado a ficheros más pequeños que son cargados desde castlib1 (para no hacer imposible la lectura). Pero realmente la organización de estos archivos de piezas no es importante porque no esperamos que nadie se ponga a estudiar las definiciones de unas piezas que, por otra parte, pueden ser cambiadas. Lo importante es la manera en que podemos utilizar estas piezas empleando las macros.

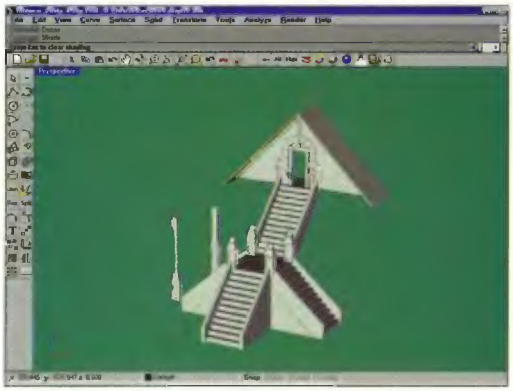
Lo que se pretende hacer

Utilizar las piezas básicas de la librería para montar casas a mano es bastante tedioso, incluso haciéndolo desde «Rhino». Es mucho más atractiva la idea de organizar las piezas según su tipo y escribir desde «POV» macros que puedan generar casas cuya forma, estilo y dimensiones dependan de los parámetros suministrados a dichas macros. Los propósitos de esta idea son múltiples. Se experimentará con la potencia de las nuevas sentencias de «POV» y nos ahorraremos bastante trabajo al montar las escenas. En el momento de escribir estas líneas faltan aún muchas cosas por probar, pero esperamos que la versión definitiva de la presente versión pueda hacer las siguientes cosas:

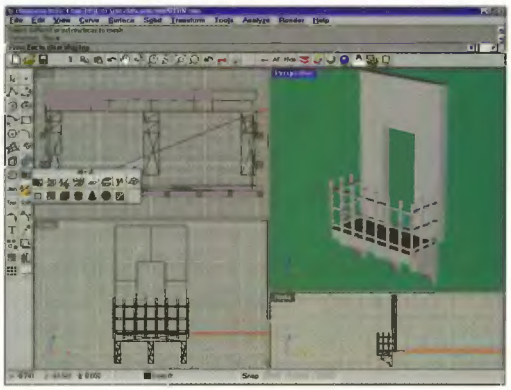
- Las casas y estructuras se montarán en una serie de estructuras situadas a diversas alturas y conectadas entre sí con escaleras. Las casas marcarán el espacio ocupado dentro de estas superficies gracias a un array que se empleará como mapa de las calles. Este mapa servirá para determinar las zonas despejadas donde pueden situarse habitantes. De esta forma, podríamos preparar una batalla en las calles de una ciudad, utilizando una macro que buscase las zonas libres del array para los combatientes. Por ahora no se contempla que este array-mapa tenga tres dimensiones (lo que serviría para mapear criaturas como dragones o grifos volando a baja altura sin miedo a que pudiesen superponerse sobre una casa o torre demasiado alta).
- Con muy pocos cambios, la librería de funciones debería poder ser utilizada para generar escenas empleando piezas básicas diferentes. De esta manera, y con muy poco esfuerzo, podríamos crear escenas de poblados, fortalezas orcas o incluso Evormigueros.
- Resulta fácil situar habitantes aprovechando la colocación de ciertas piezas de la librería. Una de ellas, por ejemplo, es un sencillo balcón que puede colocarse en los pisos donde se ha permitido que haya puertas que dan a la calle. Dando a un personaje las mismas coordenadas de dicho balcón y sumándole una pequeña traslación resulta fácil colocarlo apoyado en éste y mirando a la calle, sin que importe en qué parte de la ciudad esté. Y lo mismo cabe decir de las ventanas, escalinatas y otras piezas.
- La velocidad del render debería ser sensiblemente superior a la conseguida con «Castlib 2.0». Las pruebas realizadas hasta ahora indican que esto será así a pesar de que no se han llevado a cabo aún con pueblos grandes. Hay que decir que aunque las nuevas piezas evitan muchas superposiciones, éstas no han podido ser erradicadas totalmente. También debería producirse un ahorro de memoria de cálculo, aunque aún no se han podido realizar comparaciones definitivas en este punto.

Las macros de la librería

Las macros definitivas de «Castlib 2.5» todavía se encuentran lejos de estar acabadas en el momento de escribir estas líneas. Sin embargo, no se espera que la estructura básica de las macros ya escritas sufra cambios de importancia, como tampoco es de esperar que esto suceda con los arrays empleados para definir la estructura de los datos empleados. Básicamente las escenas se construirán empleando una única macro a la que llamaremos Trazacalles (y que todavía no está escrita). El cometido de esta macro será el de crear casas dentro de un área definida por unas coordenadas. Para ello la macro llamará a otra macro denominada MakeHouse (que sí está escrita al 90%) y ésta a su vez llamará a otras como CreaMuro y MakeTejado (que están terminadas al 90% y al 10%, respectivamente). Como ya imaginaréis por los nombres, MakeHouse crea las casas individuales de la escena empleando para ello los valores de parámetros para el estilo, las dimensiones, el número de pisos y los lados visibles. ¿Qué significa esto de los lados visibles? Bien, simplificando, todas las casas de la librería tienen únicamente cuatro paredes principales (las estructuras laterales y colgantes pueden entenderse como otras casas de menor tamaño pegadas o colgadas a la principal). Estas paredes forman planos que guardan siempre 90 grados con las paredes laterales y 180 con la pared opuesta. Esto significa que, por supuesto, siempre veremos, únicamente, dos de los lados de cualquier casa. Los lados opuestos no serán visibles y entonces, ¿para qué generarlos a ellos y a sus objetos correspondientes? Por ahora, en las pruebas, la variable "lados" que se encarga de este detalle es pasada a Makehouse manualmente. Sin embargo, próximamente su valor será puesto desde Trazacalles, dependiendo de la posición que hayamos asignado a la cámara.



Las escaleras de «Castlib 2.5».



El sencillo balcón de Castlib 2.5

En cuanto a CreaMuro, su cometido es bien simple: debe generar un muro de una longitud determinada utilizando la lista de paneles de la serie que se le indica en uno de los parámetros. Una serie es, en nuestra librería, una lista de objetos-panel definida en el array "serie_paneles". Por otra parte, el array "lista_estilos" guarda una lista de series que configuran un estilo. Al llamar varias veces a CreaMuro para levantar la casa, MakeHouse puede pasarle diferentes series como parámetro en cada ocasión. Se puede, pues, definir una serie para el primer piso, otra para los superiores, etc. (Tened en cuenta que, por ahora, hay dos construcciones básicas: la rústica y la de piedra).

Un posible problema

Un detalle que preocupa es el de la expansión del código de las macros. Para crear cada casa, «POV» debe expandir el código correspondiente con cada llamada a las macros empleadas. Esto plantea la pregunta de si desecha «POV» el código expandido una vez lo está compilando. Si esto es así (lo que es de suponer) no hay motivo para preocuparse, pero si sucede lo contrario, el consumo de memoria podría dispararse en casos como el presente. En fin, será una de las cosas que comprobaremos en el próximo numero.

NOTA INFORMATIVA

La portada, tres antiguas obras de Nathan O´Brien, no tienen nada que ver con «Castlib 2.5».

Rhinoceros: operaciones CSG y problemas con objetos

Desde su primera aparición en el número 11 de Rendermanía, «Rhinoceros» ha ido ganando rápidamente adeptos entre los lectores. Esto es comprensible ya que el programa de McNeel y Asociados es potente, cómodo, rápido y fácil de manejar. Sin embargo, los numerosos cambios e innovaciones que sus creadores han ido introduciendo con las sucesivas betas de este modelador pueden acabar despistando al lector, y por ello analizaremos la filosofía de trabajo de la última beta publicada en la dirección **www.rhino3d.com**.

En los artículos de los números 11 y 16 dedicados a «Rhinoceros» hablábamos de este programa como un modelador de splines cuyo sistema de trabajo se basaba sobre todo en el dibujo de curvas que luego podían ser "elevadas" para obtener superficies. Esta idea sigue siendo válida pero la inclusión de nuevas opciones de modelado o la mejora de otras que ya existían puede acabar cambiando la filosofía de trabajo de muchos de los usuarios de «Rhinoceros».

Cambios, modificaciones y problemas

«Rhino» es, de momento, nuestro modelador favorito y ello se debe a las siguientes razones:

- Se trabaja con mucha rapidez: la interfaz de manejo permite realizar diferentes secuencias de acciones complejas con muy pocos clicks de ratón.

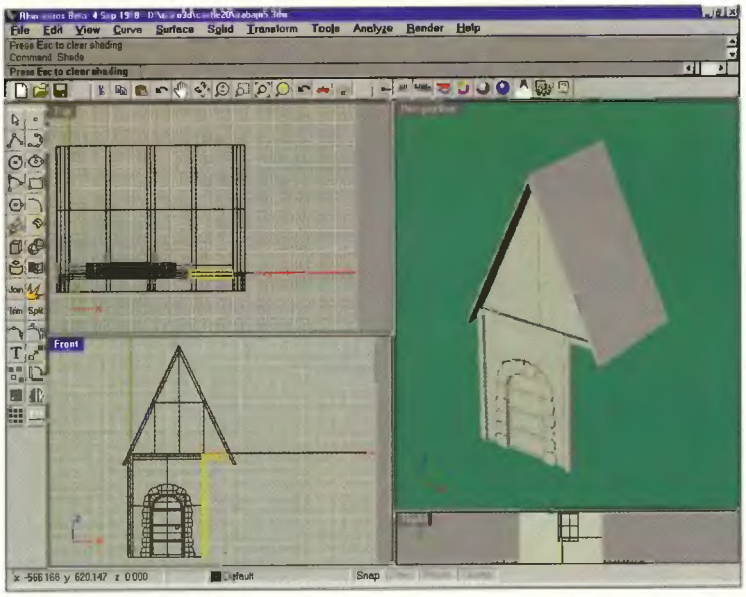


Imagen de las piezas del proyecto realizadas «con Rhinoceros».

Con «Rhino», operaciones que se eternizarían en programas antiguos como «3D Studio» o «Imagine» se llevan a cabo en muy poco tiempo.

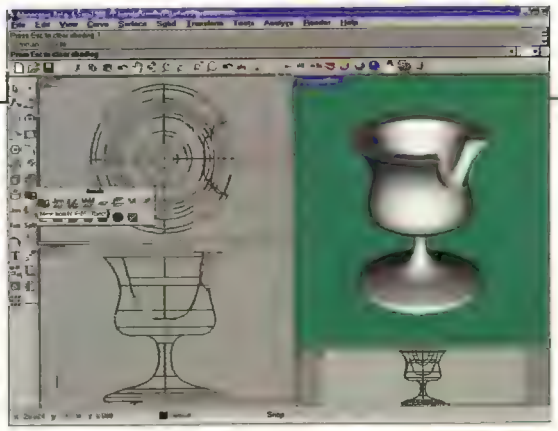
- Se trabaja con curvas spline y las opciones de elevación de estas curvas (para crear superficies) son muy potentes. Esto quiere decir que se pueden crear numerosas formas orgánicas con cierta facilidad (al menos, más fácilmente que con los antiguos modeladores poligonales).
- Las operaciones booleanas parecen muy fiables. En las pruebas realizadas por la redacción de la revista, no se encontró nunca con ningún mensaje que indicara que la operación ordenada no podía realizarse.
- «Rhinoceros» contempla la exportación de modelos para muchos otros programas, entre los que cabe destacar a «3D Studio», «Lightwave», «AutoCAD» y «POV».

Pero no todo son ventajas, aunque se rumorea que la versión definitiva de «Rhinoceros» será lanzada en octubre o noviembre, lo cierto es que la beta analizada en estas páginas (que expira el 31 de octubre) presenta algunas pegas. Esto puede parecer extraño en un producto cuya primera beta apareció hace ya dos años, pero no por ello es menos cierto que hay iconos que no funcionan, órdenes que no son reconocidas y una documentación (en la ayuda) que va muy por detrás de los cambios y de las nuevas opciones del programa. La explicación de estos problemas parece estar en la reorganización de los iconos del programa y en el cambio gráfico de casi todos ellos.

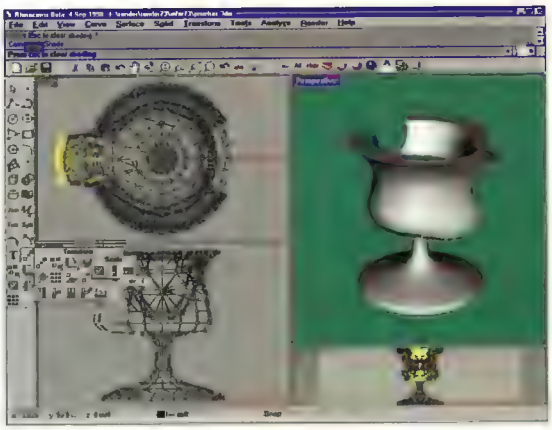
En esta versión la distribución de los iconos parece mas lógica y su representación gráfica explica mejor las operaciones relacionadas, pero en la documentación visible en la ayuda se siguen empleando los gráficos ya descartados de los iconos antiguos. En cuanto a los iconos que no funcionan, esto podría deberse a la premura con que se realizaron los cambios antes del lanzamiento definitivo que –se presume– está al caer. En fin, es de esperar que estas pegas desaparezcan en la próxima versión del programa.

Comandos de teclado

En «Rhinoceros» se puede trabajar de tres formas diferentes. Se pueden ordenar las diversas acciones pulsando en la opción adecuada del menú o sobre el icono correspondiente, o bien tecleando directamente el comando asociado. Aunque lo mas rápido suele ser utilizar los iconos, hay ocasiones donde puede resultar más rápido escribir un comando en la consola (la fila que sigue a "Command", sobre la línea horizontal de iconos). La verdad es que, de forma automática, cada vez que se pulsa un icono o se elige una opción del menú, el



Las superficies de revolución también admiten CSG...



...pero antes de editar los puntos de la copa habremos de convertirla en un objeto poligonal.

comando de teclado asociado aparece en la consola, con lo que poco a poco y sin darnos cuenta, iremos aprendiendo órdenes escritas. Además, algunas órdenes necesitan parámetros escritos por teclado. Pero al final, utilizar más o menos la consola dependerá de la forma de trabajo de cada uno. La redación suele escribir las órdenes para manipular el grid y algún que otro comando como Csec (utilizado para crear secciones a partir de curvas perfil dibujadas previamente). Veamos algunos de estos comandos:

- "gridsize": en «Rhinoceros», como en tantos otros modeladores, las ventanas de trabajo exhiben una malla de líneas llamada grid o retícula cuyas intersecciones se utilizan para facilitar el dibujo. Por ejemplo, cuando estemos dibujando curvas, los extremos de éstas se ajustarán a las intersecciones de la retícula. Y, de la misma manera, el grid resultará vital en otras operaciones como el desplazamiento, la rotación y el escalado de objetos. Pues bien, el tamaño de las intersecciones del grid puede ser establecido con el comando gridsize al que deberá seguir un valor indicando el tamaño. (Escribid gridsize y pulsad enter. «Rhino» mostrará el tamaño del grid actual. Meted el nuevo valor y pulsad enter).
- "grid": este comando activa o desactiva el grid.
- "snap": aunque el grid aparezca en las ventanas de trabajo, esto no quiere decir que lo estemos utilizando. Para hacer que los comandos de dibujo o edición se ajusten a las intersecciones del grid, teclead el comando snap que activa o desactiva el empleo del grid.
- "snapsize": aunque las intersecciones del grid tengan un tamaño establecido, el ajuste en las operaciones de dibujo o edición puede no coincidir con sus dimensiones (aunque normalmente será así). Para determinar las unidades de distancia que hay entre cada paso tendremos que emplear el comando "snapsize".
- "gridsections": este comando se utiliza para especificar el número de intersecciones del grid (y por tanto el tamaño de la malla).
- "gridthick": cada cierto número de líneas, la malla de la retícula tiene una línea de un color más claro. La utilidad de dicha línea es la de ayudarnos a medir las distancias en las diversas operaciones. Utilizando "gridthick" podremos especificar cada cuántas líneas normales queremos que aparezca una línea clara.
- "ortho": a veces puede interesarnos no emplear el grid, pero sí que el punto a dibujar se mantenga a lo largo de una línea dada o que el objeto a rotar sólo pueda hacerlo un número x de grados en cada paso. Con este comando las operaciones se harán de x en x grados.
- "orthoangle": con este comando especificaremos el número de grados que se empleará en cada paso si la opción ortho está activada (el valor por defecto es 90).

Organización y uso de los iconos

Normalmente, se emplean, preferentemente, los iconos a las órdenes de teclado o a las opciones del menu. Éstos están divididos en dos áreas. En la barra horizontal de iconos se puede acceder a las opciones de control de las ventanas de trabajo. De izquierda a derecha y a partir del icono de la mano se encuentran las operaciones de desplazamiento, rotación, zoom manual, zoom de ventana, centrado de escena y centrado de objetos seleccionados. También se encuentran aquí otros iconos de uso frecuente como el que permite elegir la vista para la ventana en curso (el del cochecito), el de selección ("All") –que se emplea para puntos, curvas, superficies y objetos–, el de visibilidad ("Hide"), el de sombreado (la

bola gris) y el de opciones. En cuanto a los iconos de creación y edición de puntos, curvas, superficies y sólidos, están localizados en la doble columna de iconos del lado izquierdo de la pantalla. En muchos casos, al mantener pulsado el botón izquierdo del ratón con el cursor sobre uno de estos iconos, puede verse cómo aparece una subventana emergente de iconos relacionados con el icono pulsado, el cual también aparecerá en la subventana. A veces, incluso, puede aparecer una segunda ventana emergente si repetimos la operación sobre un icono de la ventana anterior, pero esto no es muy frecuente. Por otro lado, si en lugar de mantener apretado el botón lo pulsamos rápidamente, el submenú no aparecerá. En lugar de esto se activará la orden correspondiente si hemos utilizado el botón izquierdo del ratón, o se desactivará (si es que esto es posible) si hemos empleado el botón derecho. (Nota: en algunos casos, apretar el botón izquierdo o el derecho del ratón tendrá un resultado diferente al mencionado. Por ejemplo, en el caso del icono de sombreado, si empleamos el botón izquierdo, el render de prueba se efectuará sólo sobre la ventana de trabajo en curso mientras que si utilizamos el botón derecho, el render se realizará en todas las ventanas). Naturalmente, esto no quiere decir que todos los iconos principales conecten con submenús. Muchos de estos iconos (de la doble columna principal) activan órdenes directamente, ya que se supone que van a ser empleados con frecuencia. Es este modo de trabajo el que determina la velocidad de trabajo del programa (junto con algunos trucos más como el hacer que se repita el último comando impartido si pulsamos con el botón derecho en la zona de las ventanas de trabajo).

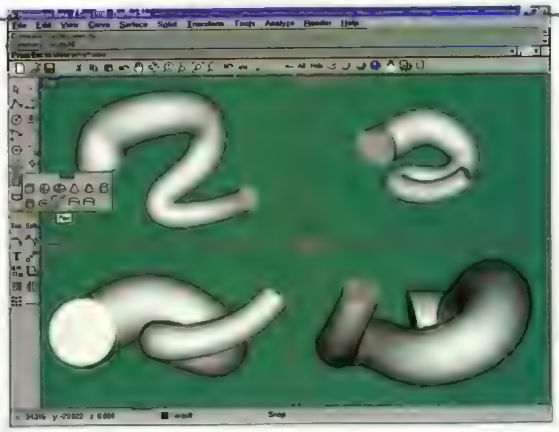
Modelado con «Rhino» por Agnisola Philippe.

El trabajo con las curvas

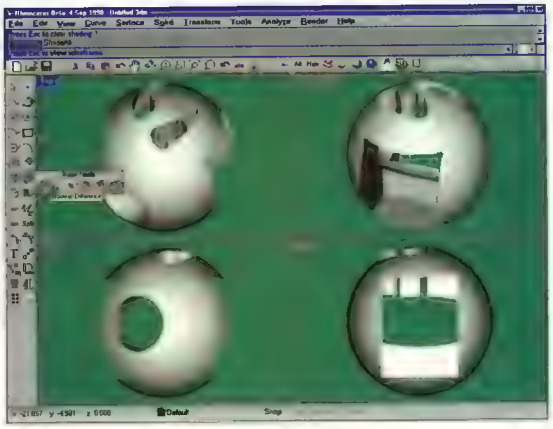
A continuación señalaremos los iconos de trabajo más empleados. Normalmente, hay que empezar por crear las curvas que luego serán empleadas como secciones a elevar o como paths (rutas) de elevación. Para crear estas curvas utilizaremos los iconos de creación de líneas rectas, curvas, círculos, elipses, arcos, rectángulos y polígonos. Todos estos iconos (situados en la parte superior de la columna principal) permiten acceder a menús emergentes de iconos, aunque normalmente nos bastará con emplear (utilizando la pulsación rápida) las opciones atadas a los iconos principales.

Una vez dibujadas las curvas podemos necesitar alterarlas, y para ello emplearemos los iconos de edición y control de puntos. Veamos cómo funcionan. Se dibuja una curva cualquiera y posteriormente se pincha sobre ella para seleccionarla (con lo que se volverá amarilla). Después habrá que pinchar sobre el icono situado bajo el llamado "Trim". Esto provocará la aparición de los puntos de edición de la curva a los que se puede pinchar y arrastrar individualmente. Además, podremos seleccionar varios puntos si pinchamos sobre ellos mientras mantenemos pulsada la tecla de shift (también podemos crear una ventana de selección si mantenemos apretado el botón izquierdo del ratón mientras arrastramos el cursor). De esta manera, podremos mover conjuntamente los puntos seleccionados (amarillos) o utilizar sobre ellos las operaciones accesibles a través del icono de transformaciones (transform), aunque no tiene sentido aplicar muchas de estas operaciones sobre los puntos de edición.

Otra posibilidad consiste en modificar la curva alterando sus puntos de control, los cuales están fuera de la curva y pueden ser editados utilizando el icono "Point editing", situado al lado del anterior y que permite acceder a una ventana emergente.



La opción pipe es ideai para construir tuberías.



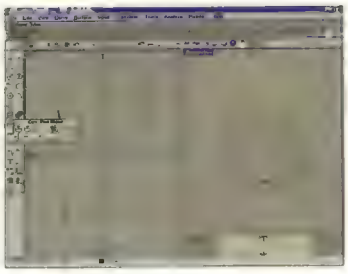
Comprobación de la fiabilidad de las operaciones CSG.

Generar superficies

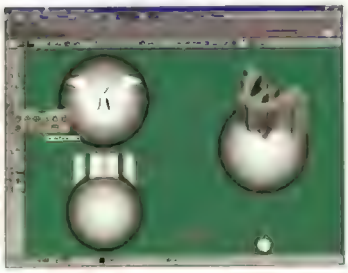
Una vez creadas las curvas y paths necesarios podemos hacer uso de las opciones de elevación accesibles a través de la ventana emergente del icono "Surface". Dichas opciones han sufrido algunos cambios desde que las estudiamos en un artículo del número 16 de Rendermanía. Aquí resumimos el propósito de las más empleadas:

- "Surface from planar Curves": con esta opción se puede llegar a generar una superficie a partir de una curva plana. Para lograr esta acción basta con seleccionar la curva y pinchar el icono.
- "Extrude Straight": permite acceder a un nuevo submenú de extrusiones de las cuales la más empleada es la propia "extrude straight" que sirve para elevar una curva a lo largo de un path recto (basta con seleccionar la curva, pinchar en el icono y luego desplazar el cursor para determinar la longitud de la extrusión). También es bastante utilizado el icono "extrude along path", con el que efectuaremos la extrusión a lo largo de una curva que habremos dibujado previamente (aunque la curva no se adapta al path, como hace "Sweep along 1 rail").
- "Loft": con esta opción crearemos una forma en tres dimensiones utilizando una serie de curvas que el programa interpretará como secciones de la forma a crear. Primero hay que dibujar las curvas y luego, tras pinchar la opción, «Rhinoceros» pedirá que se pulsen las curvas en orden. (También habrá que indicar al programa si la forma gira a lo largo del path, e igual sucede en la siguiente opción). Al terminar, se pulsará Enter y la forma será generada. Las curvas no tienen por qué tener la misma forma ni tamaño y pueden estar orientadas en distintas direcciones. Esta opción es muy utilizada para crear partes orgánicas como brazos, torsos, etc., dibujando previamente secciones de los mismos.
- "Sweep along 1 rail": el programa «Rhinoceros» pedirá que seleccionemos el patch (rail) a través del cual ha de efectuarse la elevación, y después tendremos que pulsar en orden sobre las curvas-sección que deberían haber sido dibujadas a lo largo del path. Posteriormente se pinchará sobre el icono y se pulsará F1 para ver un ejemplo de las formas que pueden lograrse con esta opción.
- "Revolve": esta es la opción que se utiliza para generar superficies de revolución. Para ello tendremos que pulsar sobre ella y posteriormente hacer lo propio sobre la curva a "revolucionar" en la ventana "Front". A continuación se indicará el eje de revolución en la misma ventana y se pulsará la tecla Enter.

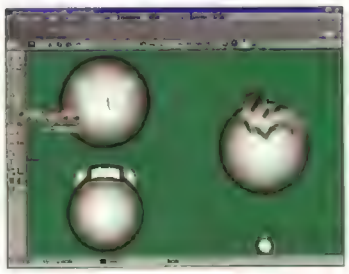
La verdad es que todas estas opciones no sólo sirven para crear superficies. En casi todos los casos es posible especificar si se quiere generar una forma cerrada o no. Utilizando "Extrude Straight", por ejemplo, basta con teclear "cap" (tapa) y pulsar Enter cuando «Rhinoceros» pida la distancia de la extrusión. Este comando de teclado activa o desactiva la colocación de tapas en la forma a elevar. En el caso de "Loft", en cambio, habrá que marcar la opción "closed loft" dentro de la ventana de opciones del comando. Es importante entender que al activar la opción de colocación de tapas, «Rhinoceros» creará un sólido y no una superficie. Algo que tendrá su importancia con las operaciones CSG.



Emplearemos la opción "extrude planar Curve" con la curva seleccionada.



Posteriormente intentaremos que el resultado se curve alrededor de la esfera.



...Y para conseguirlo lo mejor es restar un par de esferas a la forma extruida.

Sólidos y operaciones CSG

Existe un conjunto de opciones que en versiones anteriores de «Rhinoceros» tenían una importancia menor, pero que van a ganar puntos a partir de la presente versión. Se trata de las operaciones CSG (geometría de construcción de sólidos). Estas operaciones constituyen la base del modelado con «POV» y también están implementadas en programas como «3D Studio» o «Imagine». En estos últimos, no obstante, estas operaciones no son demasiado fiables. ¿Por qué? Pues porque estos modeladores trabajan con objetos compuestos por polígonos y, según parece, los algoritmos de CSG que trabajan con polígonos no son fiables al 100%. En «POV», en cambio, los objetos creados por el lenguaje de este raytracer no son poligonales sino creados mediante fórmulas, y los algoritmos CSG que operan sobre este tipo de objetos son mucho más fiables que sus equivalentes poligonales. ¿Y qué sucede en el caso de «Rhinoceros»? Dicho programa trabaja con diferentes tipos de objetos: curvas, superficies, polysuperficies, sólidos y mallas (objetos poligonales). Las operaciones CSG de «Rhinoceros» funcionan sólo con superficies, polysuperficies y sólidos, y son sorprendentemente fiables. El problema radica en que si convertimos un objeto de este tipo en uno poligonal no se podrá seguir efectuando operaciones CSG con él.

Para comprobar la teoría haremos algunos experimentos. Pulsad sobre el icono solid (el del cubo) para hacer aparecer el submenú de creación de objetos sólidos de «Rhinoceros». Aquí podemos crear una serie de objetos cuyo volumen interno está "lleno" (de ahí lo de "sólidos"). Pulsad sobre el icono de la esfera y cread una. Posteriormente repetid la operación, pero esta vez con el icono del cubo y situando el nuevo objeto de manera que se superponga parcialmente con la esfera. Ahora pulsad sobre el icono "Solid Tools" (situado junto al anterior) para hacer aparecer su menú emergente. Es aquí donde se ordenan las operaciones CSG (y otras relacionadas con éstas). Pulsad sobre el icono llamado "Boolean difference", que se emplea para hacer restas CSG (las operaciones CSG también suelen ser llamadas operaciones booleanas). Hecho esto, «Rhinoceros» nos pedirá que seleccionemos una superficie o polysuperficie. Pinchad sobre la esfera. «Rhinoceros» mostrará un nuevo mensaje pidiendo que seleccionemos la superficie o polysuperficie que se restará del primer objeto. Pulsad sobre el cubo y observad el resultado. El cubo ha desaparecido y también parte de la esfera. El objeto visible es el resultado de restar a la esfera el volumen espacial del cubo. Para visualizar mejor el resultado habrá que utilizar el icono de sombreado. De esa manera se puede comprobar que el objeto resultante sigue siendo un sólido, ya que «Rhinoceros» ha creado "tapas" en la zona restada. A todos los efectos ha sido como si los objetos hubiesen sido realmente cuerpos sólidos verdaderos y el volumen del cubo se hubiera restado del de la esfera. Naturalmente, si hubiésemos restado la esfera del cubo, el resultado hubiese sido distinto.

Y ¿cómo funciona la operación de intersección CSG? Pulsando Ctrl-Z para ordenar el undo de «Rhinoceros» y luego pinchando en el icono "Boolean Intersection" del menú emergente anterior. Hecho esto habrá que volver a seleccionar los dos objetos y comprobar que esta vez el resultado ha sido un objeto que por su forma y volumen equivale al volumen interior que compartían los dos objetos iniciales.

Una vez más hay que utilizar el undo y probar, en esta ocasión, el icono "Bo-

Modelado con «Rhino» por Jerome Desvignes.

olean Union". Los dos objetos se han convertido en uno solo (las partes internas de ambos objetos se han borrado).

La unión, la sustracción y la intersección son todas las operaciones CSG que implementa «Rhinoceros» («POV» añade una extra: merge). Con ellas es posible crear formas cuya construcción es algo más complicada utilizando superficies. Normalmente, emplearemos estas operaciones con sólidos, así que regresemos al submenú del icono de sólidos. Allí existen otros tres iconos muy interesantes. Uno de ellos es "Pipe", que es sumamente útil para modelar tubos retorcidos. Antes de utilizarlo hay que dibujar una curva abierta para que sirva como patch. Posteriormente habrá que pinchar sobre "Pipe" y seleccionar la curva. «Rhinoceros» se sitiará en uno de los extremos de la curva permitiendo definir el tamaño de un círculo que servirá como sección inicial de la extrusión. Una vez definida esta primera sección, «Rhinoceros» situará la edición en el otro extremo del path con el objetivo de crear la sección final cuyo tamaño no tiene por qué ser el mismo que el de la primera. Al hacer click el resultado deberá ser un tubo cerrado (otro sólido).

El siguiente icono de interés en el submenú de sólidos es "Extrude Planar Curve". Para estudiar su funcionamiento tendremos que crear una curva cerrada, seleccionarla, y pulsar el icono. Esto genera una extrusión de la curva a lo largo de una línea recta, tal y como hace "Extrude straight". La diferencia está en que los objetos generados mediante "Extrude Planar Curve" son sólidos, es decir, este comando crea tapas por defecto en los extremos de la nueva forma. En cuanto al siguiente icono, "Extrude Surface", funciona como el anterior y produce los mismos resultados, aunque necesita una superficie como punto de partida y no una curva.

Operaciones CSG con superficies

La forma en que «Rhinoceros» efectúa operaciones booleanas con superficies es realmente curiosa, además de útil. Supongamos que tenemos una superficie en forma de tubo abierto y una pequeña esfera que corta parcialmente el centro del tubo. En muchos modeladores, si pidiéramos al programa que restara esta esfera del tubo, la operación (en el supuesto de que el programa permitiera hacerla) daría como resultado un agujero en el tubo. «Rhinoceros», en cambio, dará otro resultado: obtendremos un abombamiento del tubo hacia adentro en la zona de resta. Esto equivale en realidad a restar el espacio de la esfera del espacio del tubo y a poner luego una tapa (con la forma de la "piel" de la esfera) en el agujero.

Pero aún hay más. Podemos convertir la superficie resultante en un sólido fácilmente. Para ello habrá que hacer aparecer el icono emergente de "Solid Tools", pulsar sobre el icono "Cap Planar Holes" y luego seleccionar el tubo del ejemplo anterior y pulsar Enter. Esto pondrá dos tapas al tubo, con lo que «Rhino» entenderá que tenemos un sólido, un cilindro, en lugar de una superficie, lo cual tendrá su efecto en las próximas operaciones CSG que se efectúen sobre el objeto. Si ahora, por ejemplo, creásemos otro cilindro algo más largo que el primero y lo colocásemos cortando el centro de aquel e hiciéramos una nueva resta booleana, el resultado sería un tubo de un determinado grosor (que seguiría siendo un sólido).

Ahora conviene tomar nota de un par de detalles. Primero, las tapas sólo pueden colocarse sobre superficies que tengan agujeros planos, como en el caso del primer ejemplo. Siguiendo con el ejemplo anterior, supongamos que pinchamos sobre el icono "Extract Surface" del menú emergente "Solid Tools". «Rhinoceros» pedirá que seleccionemos las superficies a extraer del sólido. Habrá que pinchar sobre las líneas de la resta de la esfera para seleccionar esa superficie y pulsar Enter. Podremos extraer la superficie seleccionada del cilindro que ahora volverá a ser una superficie. Pero, como el nuevo agujero no es plano, no podremos volver a cerrarlo con "Cap Planar Holes".

Utilizando operaciones booleanas es fácil crear objetos muy complejos con bastante facilidad, si, claro está, los algoritmos utilizados por el programa son fiables, y este es el caso particular de «Rhinoceros».

¿Por qué no puede hacerse esto?

Parece que el mayor problema al que suelen enfrentarse los usuarios de «Rhinoceros» es el de determinar cuándo se puede llevar a cabo o no una operación determinada sobre un objeto. Ello se debe a que en «Rhinoceros» existen diferentes tipos de objetos y a que se puede obtener una misma forma empleando diferentes caminos. Sin embargo, como todas las operaciones no funcionan con todos los tipos de objetos, habrá que tener presente en todo momento el tipo de objeto con el que se está trabajando.

Un ejemplo de esto son, como ya se ha dicho, las operaciones booleanas. Éstas no funcionan con los objetos poligonales. Sin embargo, no hay más remedio que convertir los objetos-spline en objetos poligonales para exportarlos a otros programas. Por esta razón será conveniente guardar también el fichero 3dm (el formato de «Rhino») con los objetos-spline, por si más adelante hay que hacer modificaciones. (No es posible convertir un objeto-poligonal en un objeto-spline).

Modelado con «Rhino» por Rex Sutton.

Además, tampoco se pueden editar los puntos de todos los tipos de objetos. En efecto, la edición de puntos sólo funciona con curvas, superficies y mallas poligonales, no sobre polysuperficies. Aquí el problema está en determinar cuándo el objeto es una polysuperficie o no.

Si se crea una esfera con el icono del menú de sólidos, se podrán editar sus puntos. Pero si se crea un cubo con el icono de la misma ventana, «Rhinoceros» responderá que no puede editar los puntos de una polysuperficie.

Por otra parte, si se crea un tubo sin tapas con la opción "Extrude Straight", se podrán editar sus puntos. Pero si se le añaden tapas (Cap Planar Holes), el objeto pasará a convertirse en una polysuperficie y no se podrán editar los puntos (aunque se pueden volver a editar si se le quitan las tapas). Ahora supongamos que efectuamos una operación booleana de resta sobre el tubo sin tapas. En este caso el objeto pasará a convertirse en una polysuperficie y tampoco se podrán editar sus puntos. Otro ejemplo más. Como ya hemos dicho, es posible editar los puntos de una esfera. Sin embargo, si le practicamos cualquier operación booleana, ésta pasará también a convertirse en una polysuperficie, con lo que tendremos que decir adiós a la edición de puntos. Resumiendo:

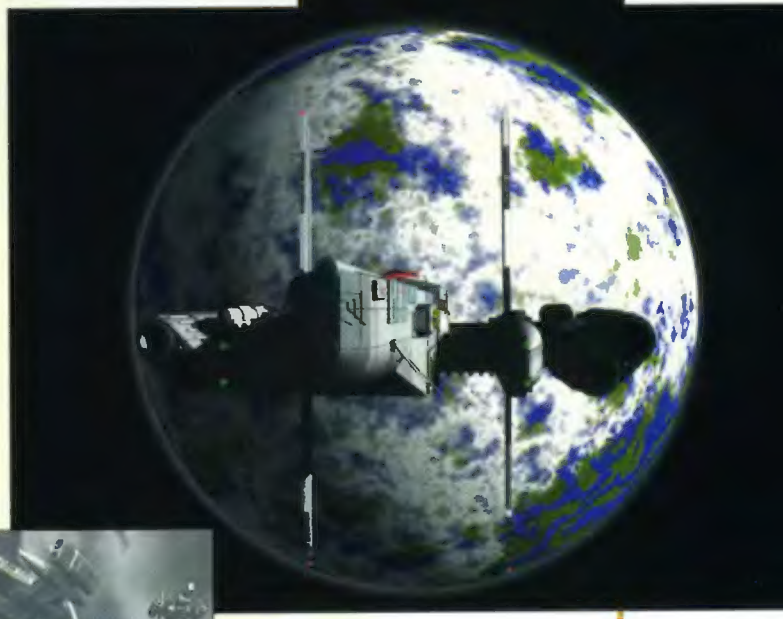
- Parece que efectuar una operación CSG sobre un objeto convierte a éste en una polysuperficie y lo mismo sucede si se le ponen tapas. Por tanto, si se quiere editar sus puntos habrá que convertirlo en un objeto poligonal.
- «Rhinoceros» no efectúa operaciones CSG sobre objetos poligonales.
- No es posible llegar a convertir un objeto poligonal a un objeto definido por curvas spline.

Estos tres puntos implican que habrá que tener sumo cuidado al pasar de un tipo de objeto a otro, porque una vez realizado no habrá vuelta atrás.

Nota importante. Podéis remitirnos vuestros trabajos o consultas, bien por carta a la dirección que figura en el sumario de PCmanía, o vía e-mail a rendermania.pcmania@hobbypress.es

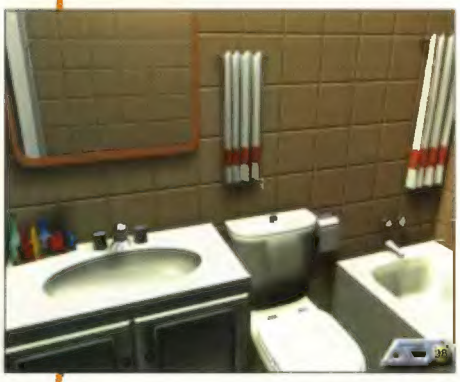
El poder del CSG

El título del foro de este número se debe a las magníficas escenas realizadas por José L. García Fuertes, quien demuestra lo que puede dar de sí la filosofía de modelado de «POV» basada en la geometría de construcción de sólidos. Pero también se podría haber basado dicho título en las preciosas escenas de interiores de Antonio Casado o en los modelos de Jesús Sáez Redondo o los de Eduardo Oliden. ¿Y qué decir de la animación de Borja Morales? Lo cierto es que todos los autores merecen una mención especial por una u otra razón.



José L. García Fuertes modela, aplica texturas y genera las escenas utilizando únicamente «POV». Tan sólo esto ya sería destacable de por sí, puesto que cada vez hay más pov-usuarios que se apoyan en modeladores externos para crear sus modelos, pero es que, además, los modelos de este autor son, por su diseño y acabado, fabulosos. El carguero estelar es probablemente la mejor nave espacial enviada nunca por un lector. Y el submarino y el galeón suponen dos grandes aportaciones al mundo medieval virtual que estamos construyendo. Y todos estos modelos han sido construidos enteramente desde «POV» Quizá se eche de menos algo de porquería en el casco del carguero y una escala mucho más pequeña para la textura de la madera en el submarino. Por lo demás, tus diseños son originales y los motelos tienen un gran nivel de detalle y un excelente acabado. Gracias por enviarnos los fuentes de estos modelos y por tu idea para la portada temática de la ciudad.

El Foro del Lector



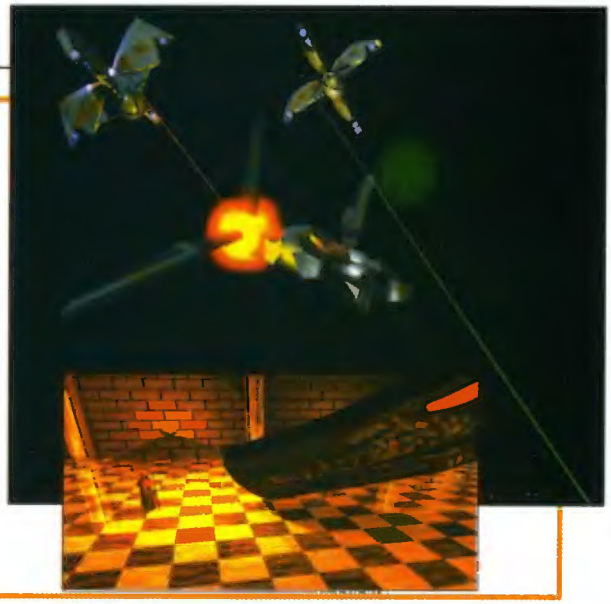
Antonio Casado es un antiguo lector que ya ha enviado varios trabajos, pero hasta ahora no había conseguido salir en Rendermanía. ¿Enviaste estos trabajos directamente a Rendermanía? En fin, tus escenas (creadas con «MAX») tienen un asombroso nivel de realismo (conseguido gracias a la impecable aplicación de unas texturas más que bien elegidas y a una excelente iluminación). Lo único que no está a la altura de todo lo demás son los botes de champú del cuarto de baño, ya que se nota enseguida que éstos son sólo una textura. En cuanto a lo que dices sobre tus animaciones, recuerda nuestros dos números anteriores dedicados a animaciones de gran tamaño. Naturalmente cuanto más pequeña sea una animación más fácilmente podrá ser publicada, pero no hay una prohibición absoluta para las películas de gran tamaño, aunque pueden tardar mucho más en aparecer. Enhorabuena por tus imágenes y también por tu página Web (cuya dirección hallaréis en el foro).

Publicamos algunos trabajos de **Juan López** enviados en varias cartas. Entre ellos hay que destacar a la simpática arañita, al coche y a su excelente R2-D2. Juan adjunta, además, los archivos de generación de algunas de estas imágenes creadas con «MAX».

Aracdhel plantea algunas cuestiones sobre las "funciones" de «POV 3.0» y las primitivas de «Rhino». La nueva sentencia #macro de «POV 3.1» admite parámetros y puede ser utilizada como una típica función de C (aunque siga trabajando expandiendo el código, como lo hacen las macros). En cuanto a la polémica polígonos vs primitivas de «POV», nos decantamos por utilizar los polígonos, aunque manipulándolos con el lenguaje de «POV» para crear estructuras mayores (empleando mesh). ¿Por qué razón? Porque normalmente importa más la velocidad que se gana con ellos que el espacio que se pierde con respecto a las primitivas de «POV».

Borja Morales (3D Reality) ha recogido el guante lanzado por la hechicera de Juan Garraza Zurbano (publicada en Rendermanía 13) y nos envía una animación creada con «3ds4» y «MAX». Ahora son los lectores quienes deben indicar con sus votos quién es el ganador. (Como recordaréis, las reglas para los combates entre personajes virtuales se indicaron en el foro del número 18 de Rendermanía). Para facilitar la votación, y ya que este es el primer duelo de este tipo que celebramos aquí, hemos incluido también la animación de Juan. En cuanto a la crítica solicitada en tu carta, lo mejor será olvidarla por una vez con el fin de no influir en la votación. Enhorabuena por tu robot creado con «Rhino» y «Metaballs» y por tu página Web.

Carlos Monterde Escudero (Erfeamor de Monticom) es un pov-adepto bien conocido en estas páginas. Este mes publicamos un antiguo trabajo suyo rescatado de un profundo pozo espacio-temporal. Se trata de un caza que Carlos ha utilizado para apalear al antiguo caza Hroki publicado en el número 3 de Rendermanía. Carlos ha incluido también un texto explicando el porqué del diseño del caza y algunos detalles más. También encontraréis aquí una demo creada por su hermano César y que os servirá para conocer mejor a este seguidor del bando orco. Carlos nos envía también una sugerencia interesante: la de escribir historias que ayuden a perfilar mejor el mundo medieval de corte fantástico que estamos creando. Tomamos nota Carlos (aunque para ser justos ya hace tiempo que hay lectores que envían historias relativas a sus escenas o personajes, como Jose M. García Lorenzo).



Eduardo Oliden es un antiguo seguidor de Rendermanía que se confiesa gravemente afectado por el virus infográfico. Este mes publicamos algunos de sus últimos trabajos con «MAX» entre los que hay que destacar al arácnido guerrero de Starship Troopers. Enhorabuena por tus excelentes modelos... y también un tirón de orejas por no enviar la malla del bichito.

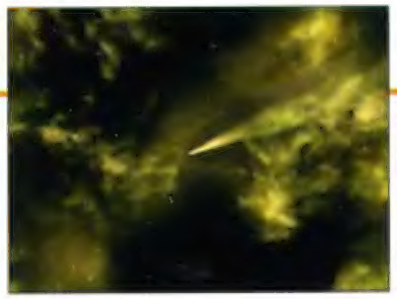
Jesús Sáez Redondo nos envía algunas imágenes de sus últimos trabajos creados empleando los programas de diseño «Metaballs», «MAX» y «Rhinoceros». Jesús pide una opinión sobre ellos: se echa en falta una mayor resolución en los bitmaps del Spitfire (los bordes de los dibujos quedan borrosos) y quizá también algo más de detalle en el revólver. Por lo demás, los modelos, sobre todo los del tanque y el Rex, son espectaculares y las texturas han sido bien elegidas. Esperamos tu próxima visita.

José M. García Lorenzo nos envía algunas sugerencias... y una pequeña reprimenda por haber confundido su nombre en el número 21. Gracias por tus sugerencias sobre las utilidades de animación. Y sí, sería preferible que todas las animaciones enviadas no excediesen los 5 Megas comprimidos (pero esto no supone una prohibición absoluta para las demás).

El Foro del Lector

José Ignacio Álvarez Ruiz (Nacho) nos envía unas pov-escenas –una de ellas generada con el pov-ipas de Colefax– y solicita a los rendermaníacos que envíen ficheros de generación y mallas para que los más novatos puedan aprender. ¡Excelente sugerencia! Por lo demás, tomamos nota de tu petición sobre lo del pov-ipas que genera teclados para «POV» (busca en el subdirectorio de Sonya, en Rendermanía).

Noé Casas Mauzanares nos remite una serie de imágenes de su cuarto creadas con «MAX». También incluye un Mech modelado en «MAX» y convertido (gracias a «Rhinoceros») en un modelo articulable y renderizable desde «POV». Sin embargo, quizá lo más destacable sea la imagen arena enteramente creada con el programa «POV», donde este autor empleó los bucles de «POV 3.0» para crear un estadio con miles de butacas. En cuanto al tiempo de cálculo, otro día explicaremos algunos truquillos para ganar tiempo en el render.



Oriol Bagau nos envía un brujo creado con «Imagine» para la próxima batalla medieval. Pero se te ha olvidado especificar el bando al que piensas apoyar, aunque hay que creer que eres un adepto de los nomuertos, ya que tu Warlock Jr. utiliza el esqueleto publicado en el número 9 de Rendermanía. En cuanto a tus preguntas, tendrán que esperar al próximo artículo que dediquemos a «Imagine», programa al que últimamente tenemos abandonado.